LE LANGAGE DE REQUETES SQL INTRODUCTION

- Origines et Evolutions
- SQL1 86: la base
- SQL1 89: l'intégrité

1. Origines et Evolutions

- SQL est dérivé de l'algèbre relationnelle et de SEQUEL
- Il a été intégré à DB2, ORACLE, MySQL, SQLServer, etc.
- Il existe trois versions normalisées, du simple au complexe :
  - SQL1 86 version minimale
  - SQL1 89 (intégrité)
  - SQL2 (92) langage complet
- Une version 3 étendue (objets, règles) est la norme 99.
- La plupart des systèmes supportent SQL2 complet

Opérations

- Opérations de base
  - SELECT, INSERT, UPDATE, DELETE
- Opérations additionnelles
  - définition et modification de schémas
  - définition de contraintes d'intégrité
  - définition de vues
  - accord des autorisations
  - gestion de transactions

Organisation du Langage

- SQL comprend quatre parties :
- Le langage de définition de schéma (Tables, Vues, Droits)
- Le langage de manipulation (Sélection et mises à jour)
- La spécification de modules appelables (Procédures)
- L'intégration aux langages de programmation

SQL1 - 86

- LANGAGE DE DEFINITIONS DE DONNEES
  - CREATE TABLE
  - CREATE VIEW
- LANGAGE DE MANIPULATION DE DONNEES
  - SELECT OPEN
  - INSERT FETCH
  - UPDATE CLOSE
  - DELETE
- LANGAGE DE CONTROLE DE DONNEES
  - GRANT et REVOKE
  - BEGIN et END TRANSACTION
  - COMMIT et ROLLBACK

2. SELECT: Forme Générale

- SELECT <liste de projection>
- FROM <liste de tables>
- [WHERE <critère de jointure> AND <critère de restriction>]
- [GROUP BY <attributs de partitionnement>]
- [HAVING <citère de restriction>]

- Restriction :
  - arithmétique (=, <, >, ≠, ≥,≤□)
  - textuelle (LIKE)
  - sur intervalle (BETWEEN)
  - sur liste (IN)
- **Possibilité de blocs imbriqués par :**
  - **IN, EXISTS, NOT EXISTS, ALL, SOME, ANY**

Forme générale de la condition

<**search condition**> ::= **[**NOT**]**

<**nom_colonne**> **θ constante** | <**nom_colonne**>

<**nom_colonne**> LIKE **<modèle_de_chaîne>**

<**nom_colonne**> IN <**liste_de_valeurs**>

<**nom_colonne**> **θ (**ALL | ANY | SOME**)** <**liste_de_valeurs**>

EXISTS <**liste_de_valeurs**>

UNIQUE <**liste_de_valeurs**>

<**tuple**> MATCH **[**UNIQUE**]** <**liste_de_tuples**>

<**nom_colonne**> BETWEEN **constante** AND **constante**

<**search condition**> AND | OR <**search condition**>

avec

θ ::= < | = | > | ≥ | ≤ | <>

Remarque: <**liste_de_valeurs**> **peut être dynamiquement déterminée par une requête**

3. Les Mises à Jour

- INSERT
  - Insertion de lignes dans une table
  - Via formulaire où via requêtes
- UPDATE
  - Modification de lignes dans une table
- DELETE
  - Modification de lignes dans une table

Commande INSERT

- INSERT INTO <relation name>
  [(attribute [,attribute] …)]
  {VALUES <value spec.> [, <value spec.>] …| <query spec.>}

Commande UPDATE

UPDATE <relation name>

SET <attribute = {value expression | NULL}

[<attribute> = {value expression | NULL}] …

[WHERE <search condition>]

Commande DELETE

- DELETE FROM <relation name>
- [WHERE <search condition>]

4. Contraintes d'intégrité

- Contraintes de domaine
  - Valeurs possibles pour une colonne
- Contraintes de clés primaires
  - Clé et unicité
- Contraintes référentielles(clé étrangères)
  - Définition des liens inter-tables

SQL1 - 89 : INTEGRITE

- VALEURS PAR DEFAUT
  - CREATE TABLE The
  - (Nthe INT UNIQUE,
  - origine CHAR(10),
  - annee INT,
  -)
- CONTRAINTES DE DOMAINES
  - SALAIRE INT CHECK BETWEEN 6000 AND 100000

SQL1 - 89 : Contrainte référentielle

- Clé primaire et contrainte référentielle
  - A discuter

SQL1 – 89 : Création de table

CREATE TABLE <nom_table>
(<def_colonne> *
[<def_contrainte_table>*]) ;

< def_colonne > ::=
<nom_colonne> < type | nom_domaine >
[CONSTRAINT nom_contrainte
< NOT NULL | UNIQUE | PRIMARY KEY |
CHECK (condition) | REFERENCES nom_table (liste_colonnes) >]

< def_contrainte_table > ::= CONSTRAINT nom_contrainte
< UNIQUE (liste_colonnes) | PRIMARY KEY (liste_colonnes) |
CHECK (condition) |
FOREIGN KEY (liste_colonnes) REFERENCES nom_table (liste_colonnes) >
[NOT] DEFERRABLE

Autre création de tables

```
CREATE TABLE EXPEDITIONS
  ( numExp INTEGER PRIMARY KEY
    date_exp DATE,
    qte QUANTITE,
    CONSTRAINT refCom FOREIGN KEY numExp
       REFERENCES  COMMANDES (numCom) DEFERRABLE
) ;
```

L'association d'un nom à une contrainte est optionnelle.

Ce nom peut être utilisé pour référencer la contrainte (ex: messages d'erreurs).

5. CONCLUSION

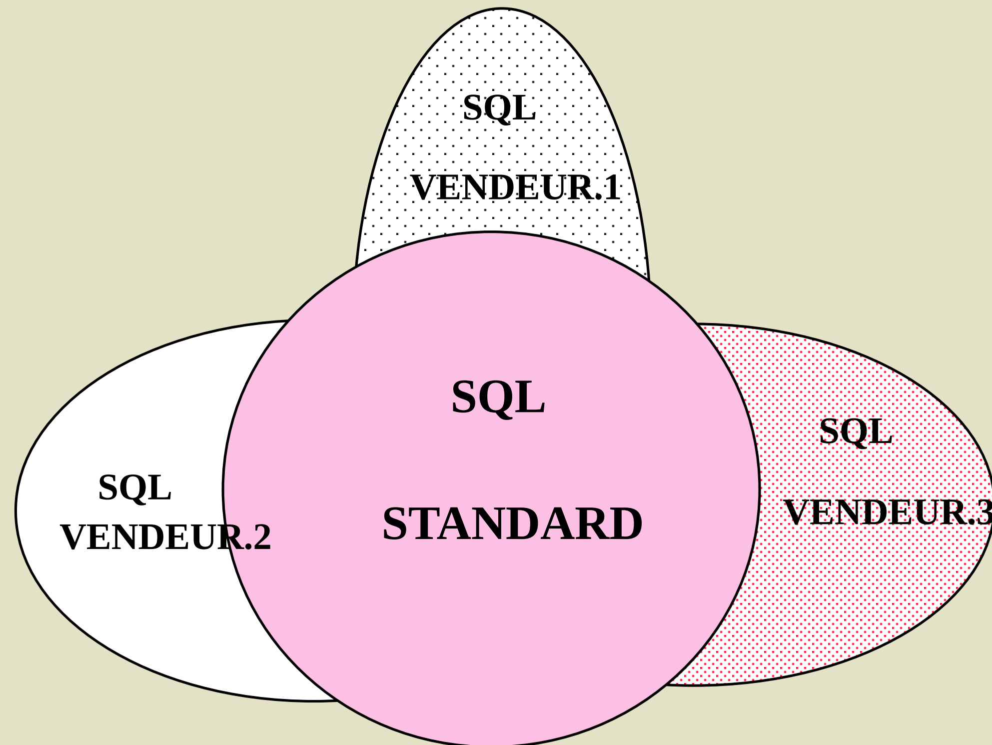
- SQL1 est un standard minimum
- Les versions étendues:
  - SQL2 = Complétude relationnelle
  - SQL3 = Support de l'objet
- Sont aujourd'hui intégrées dans les grands SGBD

LA NORMALISATION DE SQL

- Groupe de travail ANSI/X3/H2 et ISO/IEC JTC1/SC2
- Documents ISO :
  - SQL1 - 86 : Database Language SQL X3.135 ISO-9075-1987)
  - SQL1 - 89 : Database Language SQL with Integrity Enhancement X3.168 ISO-9075-1989
  - SQL2 - 92 : Database Language SQL2 X3.135 ISO-9075-1992
- Arguments pour :
  - Réduction des coûts d'apprentissage
  - Portabilité des applications
  - Longévité des applications
  - Langage de communication inter-systèmes
- Arguments contre :
  - Manque de rigueur théorique
  - Affaiblit la créativité

POSITION DES VENDEURS

- Problèmes !!!



SQL
EXTRACTION DE DONNÉES

Vue d'ensemble

- Extraction de données à l'aide de l'instruction SELECT
- Filtrage des données (Restriction)
- Mise en forme des ensembles de résultats

Extraction de données: utilisation de l’instruction SELECT

- La liste de sélection indique les colonnes
- La clause WHERE indique la condition limitant la requete
- La clause FROM indique la table

Extraction de données: Spécification des colonnes

- SELECT emp_ID, nom, prenom
- FROM employe

Extraction de données: utilisation de la clause WHERE pour specifier des lignes

- SELECT emp_ID, nom
- FROM employe
- WHERE empl_ID=5

Filtrage des données

- Utilisation des opérateurs de comparaison
- Utilisation des comparaisons de chaines
- Utilisation des opérateurs logiques
- Extraction d'une plage de valeurs
- Utilisation d'une liste de valeurs comme critère de recherche
- Extraction de valeurs inconnues

Filtrage des données

Type de filtre	Condition de recherche
Opérateurs de comparaison	=,<,>,<=,>= et <>
Comparaison de chaines	LIKE et not LIKE
Opérateurs logiques	AND, OR, NOT
Plage de valeurs	BETWEEN et NOT BETWEEN
Liste de valeurs	IN et NOT IN
Valeur inconnues	IS NULL et IS NOT NULL

Filtrage des données: Utilisation des opérateurs de comparaison

- SELECT prenom, ville
- FROM employe
- WHERE pays=‘MAROC’

Filtrage des données: Utilisation des comparaison de chaines

- SELECT nom_entreprise
- FROM client
- WHERE nom_entreprise LIKE '%MAROC%'

Filtrage des données: Utilisation des opérateurs logiques

- SELECT produit_ID, produit_nom
- FROM produit
- WHERE (produit_nom LIKE 'toto' OR produit_ID=20) AND (PU_HT>100)

Filtrage des données: Extraction d'une plage de valeurs

- SELECT produit_nom, PU_HT
- FROM produit
- WHERE PU_HT BETWEEN 10 AND 20

Filtrage des données: Utilisation d'une liste de valeurs comme critère de recherche

- SELECT nom_entreprise, pays
- FROM fournisseurs
- WHERE pays IN ('MAROC','ALGERIE')

Filtrage des données: Extraction de valeurs NULL

- SELECT nom_entreprise, fax
- FROM fournisseurs
- WHERE fax IS NULL

Mise en forme des ensembles de résultats

- Tri des données
- Suppression des doublons
- Changement des noms de colonne
- Utilisation de litteraux

Mise en forme :
Tri des données

- SELECT produit_ID, nom_produit, categorie_ID
- FROM produit
- ORDER BY categorie_ID

Mise en forme : Suppression des doublons

- SELECT DISTINCT pays
- FROM fournisseurs
- ORDER BY PAYS

Mise en forme : Changement des noms de colonne

- SELECT nom as N, prenom as P, employe_ID as ‘EMPLOYE ID’
- FROM employe

Mise en forme : Utilisation de litteraux

- SELECT nom, ‘NUMERO IDENTIFICATION :’ employe_ID
- FROM employe

SQL

Regroupement

Utilisation de fonctions d’agrégation

Fonction d’agrégation	Desciption
AVG (expr)	Moyenne de expr
COUNT (* | expr)	Nombre de ligne
MAX (expr)	Maximum de expr
MIN (expr)	Minimum de expr
SUM (expr)	Somme de exp
STDEV (expr)	Ecart type de exp
VAR IANCE(expr)	Variance de exp

Utilisation de fonctions d'agrégation

- Exemple

```
Select SUM(Quantite)
From detail_commande
```

Utilisation de la clause GROUP BY

- Select produit_ID, Commande_ID, Quantite
- From detail_commande

- Select produit, SUM(Quantite)
- From detail_commande
- GROUP BY produit_ID

Utilisation de la clause GROUP BY

Produit_ID	Commande_ID	Quantite
1	1	5
1	4	10
1	7	20
2	1	5
2	2	10



Produit_ID	Total_Qte
1	35
2	15

GROUP BY avec Having

- SELECT produit_ID, SUM(Quantite),
- From detail_commande
- GROUP BY produit_ID
- HAVING SUM(Quantite) >30

MANIPULATION DE DONNÉES

Objectif

Décrire l'aspect LMD (langage de manipulation des données) de MySQL. Nous verrons que SQL propose trois instructions pour manipuler des données :

- l'insertion d'enregistrements : INSERT ;
- la modification de données : UPDATE ;
- la suppression d'enregistrements : DELETE (et TRUNCATE).

Il existe d'autres possibilités pour insérer des

données en utilisant des outils d'importation ou de migration, citons MySQL Migration Toolkit, SQLPorter, Navicat, Intelligent Converters et MySQL Data Import de la société EMS.

Insertions d’enregistrements (INSERT)

Insertions d’enregistrements (INSERT)

- Pour pouvoir insérer des enregistrements dans une table, il faut que vous ayez reçu le privilège INSERT. Il existe plusieurs possibilités d’insertion : l’insertion monoligne qui ajoute un enregistrement par instruction et l’insertion multiligne qui insère plusieurs enregistrements par une requête.

Renseigner toutes les colonnes

INSERT INTO Compagnie
VALUES ('SING', 7, 'Camparols', 'Singapour', 'Singapore AL');

INSERT INTO Compagnie
VALUES ('AC', 10, 'Gambetta', **DEFAULT**, 'Air France');

Renseigner certaines colonnes

- Insérons deux lignes dans la table Compagnie en ne précisant pas toutes les colonnes. La première insertion affecte implicitement la valeur par défaut à la colonne ville. La deuxième donne implicitement la valeur NULL à la colonne nrue.

```
INSERT INTO Compagnie(comp, nrue, rue,
nomComp)
VALUES ('AF', 8, 'Champs Elysées', 'Castanet
Air');
```

Plusieurs enregistrements

Le script suivant ajoute trois nouvelles compagnies en une seule instruction INSERT.

```
INSERT INTO Compagnie VALUES
('LUFT',9,'Salas','Munich','Luftansa'),
('QUAN',1,'Kangouroo','Sydney','Quantas'),
('SNCM',3,'P. Paoli','Bastia','Corse Air');
```

Énumérations

Le type ENUM est considéré comme une liste de chaînes de caractères. Toute valeur d’unecolonne de ce type devra appartenir à cette liste établie lors de la création de la table. Supposons qu’on recense quatre types possibles de diplômes ('BTS', 'DUT', 'Licence' et 'INSA') pour chaque étudiant. On ne stocke qu’un seul diplôme par étudiant.

```
CREATE TABLE UnCursus
(num CHAR(4), nom CHAR(15), diplome
ENUM ('BTS','DUT','Licence','INSA') ,
CONSTRAINT pk_Cusus PRIMARY KEY(num));

INSERT INTO UnCursus VALUES
('E1', 'F. Brouard', ('BTS'));

INSERT INTO UnCursus VALUES
('E2', 'F. Degrelle', 'Licence');
```

Dates et heures

Les types suivants permettent de stocker des moments ponctuels (dates, dates et heures, années, et heures). Les fonctions NOW() et SYSDATE() retournent la date et l’heure courantes. Dans une procédure ou un déclencheur SYSDATE est réévaluée en temps réel, alors que NOW désignera toujours l’instant de début de traitement

Dates et heures

type	commentaire
DATE	Sur 3 octets. L'affichage est au format 'YYYY-MM-DD'.
DATETIME	Sur 8 octets. L'affichage est au format 'YYYY-MM-DD HH:MM:SS'.
YEAR	Sur 1 octet ; l'année est considérée sur 2 ou 4 positions (4 par défaut). Le format d'affichage est 'YYYY'.
TIME	L'heure au format 'HHH:MM:SS' sur 3 octets.
TIMESTAMP	Estampille sur 4 octets (au format 'YYYY-MM-DD HH:MM:SS') ; mise à jour à chaque modification sur la table.

Dates et heures

CREATE TABLE Pilote

(brevet VARCHAR(6), nom VARCHAR(20), dateNaiss **DATE**,

nbHVol DECIMAL(7,2),dateEmbauche **DATETIME**, compa VARCHAR(4),

CONSTRAINT pk_Pilote PRIMARY KEY(brevet));

Modifications de colonnes UPDATE

Modifications de colonnes UPDATE

L’instruction UPDATE permet la mise à jour des colonnes d’une table. Pour pouvoir modifier des enregistrements d’une table, il faut que cette dernière soit dans votre base ou que vous ayez reçu le privilège UPDATE sur la table.

Modification d’une colonne

Exemple :

Modifions la compagnie de code 'AN1' en affectant la valeur 50 à la colonne nrue.

UPDATE Compagnie SET nrue = 50 WHERE comp = 'AN1';

Modification de plusieurs colonnes

- Exemple:

UPDATE Compagnie SET nrue = 14, ville = **DEFAULT** WHERE comp = 'AN2';

UPDATE Pilote SET dateNaiss = '1967-03-25 **12:35:00‘**
WHERE brevet = 'PL-1'

Suppressions d’enregistrements

Instruction **DELETE**

Les instructions DELETE permet de supprimer un ou plusieurs enregistrements d’une table. Pour pouvoir supprimer des enregistrements dans une table, il faut que cette dernière soit dans votre base ou que vous ayez reçu le privilège DELETE sur la table.

DELETE FROM Pilote WHERE compa = 'AF';

DELETE FROM Compagnie WHERE comp = 'AF';

Intégrité référentielle

Intégrité référentielle

L'intégrité référentielle forme le coeur de la cohérence d'une base de données relationnelle. Cette intégrité est fondée sur la relation entre clés étrangères et clés primaires (ou candidates : colonnes indexées uniques et non nulles) qui permettent de programmer des règles de gestion.

Ce faisant, la plupart des contrôles côté client (interface) sont ainsi déportés côté serveur.

!!!! A continuer…

SQL

Jointure
(Exemples)

Jointure naturelle

- Utilisation d’alias pour les noms des tables

- SELECT nom_fournisseur, t.id_f, qte
- FROM fournisseurs AS f INNER JOIN transaction AS t
- ON f.id_f=t.id_f

Jointure externe

- SELECT nom_fournisseur, t.id_f, qte
- FROM fournisseurs AS f LEFT OUTER JOIN transaction AS t
- ON f.id_f=t.id_f